Revealing Multiple Layers of Hidden Community Structure in Networks*

Kun He† Sucheta Soundarajan‡ Xuezhi Cao§ John Hopcroft¶ Menglong Huang‖

Abstract

We introduce a new conception of community structure, which we refer to as *hidden community structure*. Hidden community structure refers to a specific type of overlapping community structure, in which the detection of weak, but meaningful, communities is hindered by the presence of stronger communities. We present Hidden Community Detection (HICODE), an algorithm template that identifies both the strong, dominant community structure as well as the weaker, hidden community structure in networks. HICODE begins by first applying an existing community detection algorithm to a network, and then removing the structure of the detected communities from the network. In this way, the structure of the weaker communities becomes visible. Through application of HICODE, we demonstrate that a wide variety of real networks from different domains contain many communities that, though meaningful, are not detected by any of the popular community detection algorithms that we consider. Additionally, on both real and synthetic networks containing a hidden ground-truth community structure, HICODE uncovers this structure better than any baseline algorithms that we compared against. For example, on a real network of undergraduate students that can be partitioned either by 'Dorm' (residence hall) or 'Year', we see that HICODE uncovers the weaker 'Year' communities with a JCRecall score (a recall-based metric that we define in the text) of over 0.7, while the baseline algorithms achieve scores below 0.2.

1 Introduction

We propose a fundamentally new paradigm of **hidden community structure**, which is characterized by the presence of multiple **layers** of community structure. In this conception of community structure, networks contain a single, **dominant** set of strong communities, which interferes with the accurate detection of weaker, but still meaningful **hidden** community structure. We present an algorithm to detect such hidden structure, and demonstrate that real networks often contain many high-quality hidden communities that are undetected by existing community detection algorithms.

This paper is intended to introduce the idea of hidden community structure and justify its validity, and it is our hope that we spark a new line of research in community detection.

Definitions A **layer** corresponds to a division of the network into communities, defined in accordance to some metric or algorithm. If a partitioning algorithm is considered, then a single layer represents a partitioning of the network. The strongest layer, or **dominant layer**, is the layer with the strongest community score, as measured or detected by that metric or algorithm (e.g., the partitioning with the highest modularity score, or the partition that is actually detected by a modularity maximization heuristic algorithm). A **hidden layer** is a layer with a quality score that is equal to or lower than that of the dominant layer, but still represents meaningful community structure (i.e., it has more structure than one would expect in a random graph). An algorithm that partitions the network into disjoint communities will only detect the dominant layer, and an algorithm to find overlapping communities can only find hidden layers if the member communities are sufficiently strong that their structure can be detected along with the structure of the stronger layers.

Traditional community detection algorithms, whether detecting disjoint or overlapping communities, identify the strongest, clearest community structure in the network: the dominant layer. As we will show, real networks contain a great deal of significant, hidden community structure in addition to this dominant layer. Moreover, for applications in a large variety of scientific disciplines, the hidden structure may often be of greater interest. For example, to a biologist attempting to locate genes that play similar functions, the dominant structure is likely already well known. Similarly, if a sociologist wishes to find groups in society, the groups detected by a typical community detection algorithm are the strongest communities (e.g., those based on family), but are also likely to be easily obtained from other sources (such as civil records). In both cases,

*Supported by US Army Research Office W911NF-14-1-0477, and National Science Foundation of China 61472147.

†Huazhong University of Science and Technology, brooklet60@hust.edu.cn.

‡Rutgers University, s.soundarajan@cs.rutgers.edu.

§Shanghai Jiaotong University, a9108g@gmail.com.

¶Cornell University, jeh@cs.cornell.edu.

‖Huazhong University of Science and Technology, 85885453@qq.com.

there is likely to be additional meaningful community structure.

In such situations, the strongest communities may often be identified by direct non-algorithmic means, such as surveys or existing annotation. If the researcher has resorted to using an algorithm to identify not-yet-known community structure, then he or she may be especially interested in this slightly weaker, hidden community structure; and yet, existing community detection algorithms will only identify the clear and obvious dominant structure![1] Despite its importance, the concept of hidden community structure has received little attention in the research literature.

One of our major contributions is to demonstrate that across diverse networks from very different domains, many high-quality community layers (e.g., high modularity partitions) are consistently hidden from all of the existing community detection algorithms that we consider.

Hidden Structure vs. Overlapping Structure Our notion of hidden structure reflects a specific type of overlapping community structure that has not been previously explored. To better understand why traditional algorithms for detecting overlapping structure fail to identify hidden structure, consider the following: Suppose that we are given a quality function that measures how 'community-like' a set of nodes within a network is.[2] If one is given sufficient computing power, then to find communities one could consider every set of nodes from the network and calculate that set's quality score. If a set of nodes receives a score higher than one would have expected in a random graph, then it has meaningful structure and is considered to be a community. Because iterating over the power set of nodes is infeasible in practice, community detection algorithms will often use heuristics to identify sets of high-scoring communities. Typically these heuristics will correctly locate the highest-scoring communities, while communities with quality scores that are lower, but still meaningful, are left undetected.

A traditional algorithm may fail to identify hidden communities for many reasons, most of which depend on the specific algorithm being considered. We have, however, observed one key reason why most algorithms fail to locate hidden structure: If a community is mostly or completely contained within one or more communities with higher quality scores, then the algorithm will likely be unable to 'see' the weaker community behind the stronger communities.

Motivating Example Suppose that a government wishes to identify communities of a criminal organization within a social network. This structure may be much weaker than other community structures, such as those based on family or location. One would expect that the structure of the criminal organization, though weak, is expressed within the network, but algorithms may have difficulty locating it. The structure of the criminal organization is hidden, but important, and identifying it in the presence of the stronger community structures is a challenge.

We present the Hidden Community Detection algorithm template, or HICODE, a method that identifies hidden community structure in networks by iteratively removing the structure of detected layers.

Application of HICODE allows us to make three important conclusions. (1) Real networks from a wide variety of different domains contain multiple, non-redundant layers of community structure. These hidden layers often have high modularity scores, indicating that they represent meaningful structure. On the networks that we considered, up to 6 layers are found. (2) Popular community detection algorithms are unable to find most of this community structure. This occurs because the structure of the hidden layers is obfuscated by that of the dominant layer. (3) Equally importantly, the structure of the hidden layers interferes with accurate detection of the dominant layer. Most of the comparison community detection algorithms that we consider are able to detect community structure that roughly corresponds to the strongest community layer, but we can obtain a more accurate, refined representation of this layer by removing the effects of the hidden layers.

We additionally consider a case study on a real network of undergraduate students. On this network, the layers found by HICODE correspond neatly with two categories of ground truth communities: 'Year', and 'Dorm', both of which are discovered with JCRecall scores over 0.7.[3] The dominant 'Dorm' layer is represented strongly in the network structure, and most algorithms can detect it, but only HICODE can locate the weaker 'Year' partition.

The major contribution of this paper is the introduction of the notion of hidden community structure. In support of this argument, our contributions are as follows:

- We observe that real social networks contain multiple meaningful layers of communities.

[1] There are clearly many situations in which detection of the dominant layer is desired; we are arguing for the notion of hidden community structure as a complement to, rather than a replacement form, traditional conceptions.

[2] An example of such a function is conductance, which measures the ratio of external links to internal links.

[3] JCRecall is a performance metric based on recall, and can be interpreted similarly. We define it formally in Section 5.

- We show that the structure of the weaker layers may be obfuscated by that of the stronger communities, making their discovery difficult. Similarly, accurate discovery of the stronger communities is hindered by the presence of the weaker structures.
- We present HICODE, a general template for identifying multiple layers of community structure. On real and synthetic data, our algorithm identifies meaningful layers of communities that other popular methods cannot detect. Additionally, we provide a case study demonstrating that the hidden layers found by HICODE correspond to important ground truth communities.

This paper is organized as follows. Section 2 discusses related work. Section 3 describes HICODE in detail. Section 4 describes our datasets, and Section 5 contains experiments conducted on both synthetic and real data, and we conclude in Section 6.

2 Related Work

Community detection algorithms can be roughly grouped into those that partition the set of nodes in a network and those that find overlapping communities [4]. Our work complements these concepts by introducing the notion of hidden structure, in which stronger community layers obscure deeper, but still meaningful, community structure.

2.1 Algorithms for Finding Disjoint Communities A popular community metric is the modularity score, which measures the quality of a partitioning. It is defined as the ratio of the number of edges that are in the same community to the expected number of edges in the same community if the edges had been distributed randomly while preserving degree distribution [11].

The modularity Q of a partition is defined as:

$$Q = \sum_{i=1}^{c} \left(\frac{e_{ii}}{m} - a_i^2\right) \tag{2.1}$$

where c is the number of communities, m is the number of edges in the graph, e_{ii} is the number of edges within community i, and $a_i = 2 * e_{ii} + e_{ij}$, where e_{ij} is the number of edges with one endpoint in community i.

The Louvain method is a heuristic algorithm for modularity maximization that builds a hierarchy of communities by first optimizing modularity locally and then grouping small communities together into larger communities [2].

Other algorithms use random walks [5, 9, 15], with the intuition that a good community is a set of nodes that random walks tend to get 'trapped' in. One such algorithms is Walktrap, which calculates a random walk-based distance measure between every two nodes, and then clusters using these distances [13]. Another such algorithm is Infomap, which finds clusters by minimizing the expected length of a description of information flow [14].

2.2 Algorithms for Finding Overlapping Communities The Link Communities method was one of the first to approach the problem of finding overlapping communties. This algorithm calculates the similarity between adjacent edges and then clusters the links [1].

Many algorithms identify communities by expanding 'seeds' into full communities [12, 16]. Two examples are OSLOM [7], which uses nodes as seeds and joins together small clusters into statistically significant larger clusters, and Greedy Clique Expansion, which uses cliques as seeds and expands to optimize for a local fitness function based on the number of internal and external links [8].

2.3 Edge-Removal Algorithms Many algorithms remove edges from the network, and then define a community as a connected component left after the appropriate edges have been removed. Two such algorithms are the classic Girvan-Newman algorithm, which removes edges with high betweenness [6], and the more recent work of Chen and Hero, which removes edges based on local Fiedler vector centrality [3].

Young, et al. present a cascading algorithm that identifies communities by using an existing community detection algorithm, and then removes all edges within communities [18]. This process is repeated several times. We call this method 'Cascade.'

HICODE is superficially similar to Cascade, which also uses an existing method to repeatedly find community structure. Our work differs from theirs in two important ways. First, they eliminate the structure of detected layers by removing all intra-community edges. This method is harsher, and meaningful structure may be lost. In contrast, we present several more nuanced methods for edge removal. Second, their algorithm lacks the refinement stage of HICODE, in which the effects of weaker layer are removed so that the stronger layers can be more accurately discovered. We show that these two elements are key components of HICODE's success, and that HICODE far outperforms this method.

3 Proposed Method: HICODE

The primary goal of our work is to demonstrate that real networks from a variety of domains contain multiple layers of hidden community structure. This structure is meaningful, but is typically not detected by traditional community detection algorithms.

Recall some important definitions: A **layer** is a

set of communities. In this paper, a layer corresponds to a partitioning of the nodes; however, in future work, it could also correspond to a set of overlapping communities. The **dominant layer** is the strongest layer in the network as measured by some metric or detected by some algorithm. The dominant layer corresponds to the set of communities output by some algorithm when applied directly to the network. A **hidden layer** is a layer that is different from the dominant layer but also achieves a high quality score, as measured by the same metric. Detection of the hidden layers is challenging, because their structure may be obscured by that of the dominant layer. We show that real networks contain many such hidden layers.

To reveal hidden layers in networks, we propose the community detection template Hidden Community Detection, or HICODE. The HICODE template contains two basic processes: **Identification** and **Refinement**.

The **Identification** stage contains two steps. First, an existing community detection method (the 'base' algorithm) is applied to the network. The structure of the resulting communities is then reduced, and the effect of their structure on the network is removed. The hidden structure lying beneath these communities is then revealed. This process is then repeated until the correct number of layers has been identified.

The **Refinement** step consists of iterating through each detected layer L, reducing the structures of *all other* layers, and then applying the base algorithm to the resulting network structure. In contrast, the Identification step only reduces the structures of the stronger layers. Although reducing only the stronger layers is sufficient to detect layer L, weaker layers can also have an effect on how accurately it is detected. By reducing the effects of stronger and weaker community layers, a more accurate version of L is produced.

HICODE can automatically select the correct number of layers. This is accomplished by increasing the number of layers until a stopping condition is met. This aspect of HICODE is described in Section 3.3.

For the **Identification** and **Refinement** steps described below, fix the number of layers at n_L.

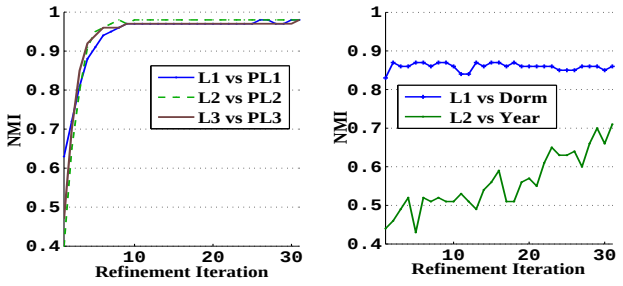
3.1 Identification The identification step is a key part of HICODE. In this step, we iteratively find layers by reducing the structures of the different community layers, where a layer is defined as the community structure that can be detected by the base algorithm being used. We experimented with several reducing methods. For each of these methods, one reduces a single layer of community structure as follows:

- The **RemoveEdge** method simply removes all intra-community edges. This method may sometimes be too coarse and destroy too much structure (particularly when communities overlap), but we include it for the sake of comparison.
- The **ReduceEdge** method randomly removes edges within each community so that the probability of seeing an edge in the community matches the background probability of seeing an edge in the network. Specifically, for a community C: Let n be the total number of nodes in the network, let e be the total number of edges in the network, let n_C be the number of nodes in community C, and let e_C be the number of edges in C. Suppose p_C is the probability that an edge exists between two randomly chosen nodes from C, so $p_C = \frac{e_C}{0.5*n_C*(n_C-1)}$. Let q_C be the background edge probability, defined as $q_C = \frac{e-e_C}{0.5*(n*(n-1)-(n_C*(n_C-1)))}$. Define $q'_C = \frac{p_C}{q_C}$. Then each edge within community C is removed with probability $1 - q'_C$; that is, edges are removed from C until the probability of an edge within C matches the background edge probability. Note that this algorithm is non-deterministic due to the randomness of the edges removal;
- The **ReduceWeight** method reduces the weight of each edge within the community C by a factor of q'_C, which is defined as in **ReduceEdge**. Like with **ReduceEdge**, we wish to make the weighted probability of an edge within C equal to the background probability of an edge, but this method is deterministic. This method is only valid if the base algorithm supports weighted networks.

Detecting and reducing community layers is repeated until n_L layers have been found.

3.2 Refinement Once the layers are identified, they are refined. In this step, for each layer, the effects of *all other* layers are reduced, and in this way we obtain the *reduced graph*. The base community detection algorithm is applied to this reduced graph. This is in contrast to the Identification step, where only the layers found so far (stronger layers) are reduced. Here, because all layers have been identified, we can reduce the weaker layers as well. This is necessary because those weaker layers can still impair detection of the layer currently under consideration, even though they have a smaller effect on the network structure than the stronger layers.

Figure 1(a) illustrates the importance of the refinement step on the SynL3 synthetic network, which contains three layers of planted communities (described further in Section 4). We applied HICODE:Mod to this network, an implementation of HICODE that uses the Louvain method for greedy modularity optimization as the base algorithm. We use the ReduceWeight reduction method, and conduct 30 iterations of refinement. The



(a) SynL3 (b) UGrad

Figure 1: Similarity (in NMI) between layers found by HICODE:MOD:REDUCEWEIGHT (layers denoted by L1, L2, L3) and ground truth community layers on synthetic network **SynL3** (denoted PL1, PL2, PL3) and real dataset **UGrad** (denoted Dorm and Year). The x-axis is the number of refinement iterations; x=0 represents results after identification, before refinement. Similarity between detected and true layers increases dramatically with refinement.

plot shows the Normalized Mutual Information (NMI) between the first, second, and third detected layers and the first, second, and third planted layers. We see that in every case, the NMI increases as more refinement is performed, indicating that the detected layers become more like the true planted layers.

Figure 1(b) contains results on the real UGrad dataset, which is described in Section 4. This dataset contains two ground truth layers, the 'Dorm' layer and the 'Year' layer. As before, we show the NMI between the first and second detected layers and the two ground truth layers. Refinement does not affect the 'Dorm' layer, but the 'Year' layer improves substantially.

In this paper, we run 30 refinement iterations. In each iteration, we take the set of layers found in that iteration and calculate the average modularity of those layers in the reduced graph. The final output is the set of layers with the highest average modularity in the reduced graph.

3.3 Selecting the Number of Layers HICODE automatically selects the number of layers by considering increasing numbers of layers until a stopping condition is met. The stopping condition uses modularity, which measures the quality of a partition. For each considered number of layers i, we calculate four values:

- $orig_0^i$ and red_0^i: These are the average modularity of all detected layers after identification, before any refinement is conducted. These values are calculated, respectively, in the original graph and in the reduced graph.[4]
- $orig_5^i$ and red_5^i: After each of the first 5 refinement iterations, we calculate the average modularity of the detected layers in the original graph. We obtain 5 averages, one for each of the first 5 refinement iterations. $orig_5^i$ is the average of these 5 values. red_5^i is similar to $orig_5^i$, but calculated in the corresponding reduced graph (e.g., for each specific layer, we first reduce the other detected layers, and then calculate the modularity of the layer being considered). These values measure how much refinement is improving the results.

One can use values other than 5 to calculate $orig_5^i$ and red_5^i, but we have found that 5 iterations of refinement is sufficient to for this purpose.

We then define $\delta_i = \frac{orig_5^i}{orig_0^i}$ and $\delta_i' = \frac{red_5^i}{red_0^i}$.

We calculate δ_i and δ_i' for the number of layers $i = 2, 3, ...$ and stop at the earliest layer L_i such that either $\delta_{i+1} < 1$ or $\delta_i' > \delta_{i+1}'$.

δ_i represents how much refinement improves the detected layers in the original graph. If the number of layers is too high, then when we remove the structure belonging to the extra layers, we will be removing structure that actually belonged to some other layer. This will result in a lower quality partitioning, such that the refinement stage actually *lowers* the quality of the detected layers. Thus, if δ_i is below 1, so $orig_5^i < orig_0^i$, then the modularity after refining is worse than if we had done no refining, so we have found too many layers and should stop.

δ_i' represents how much refinement improves the quality of the detected layers in the reduced graph. As we consider progressively higher numbers of layers, this ratio rises, as structure that was interfering with accurate layer detection is removed. When δ_i' drops, this indicates that we are removing too much structure, so we have found too many layers and should stop.

4 Datasets

We will demonstrate that real networks from varied domains contain multiple layers of hidden community structure. In our experiments, we consider real networks from social and biological domains, as shown in Table 1.

- **Grad** and **UGrad**: These networks are subsets of the Facebook network corresponding to graduate and undergraduate students at Rice University [10].
- **HS, SC, DM**: The **HS**, **SC**, and **DM** networks are genetic networks describing interactions between proteins in, respectively, *Homo sapiens*, *S. cerevisiae* (a yeast), and *D. melanogaster* (a fruit fly).

[4] Recall that the reduced graph is obtained by reducing the structure of all layers other than the one being considered.

Network	# Nodes	# Edges	Diameter	Clustering Coef.
Grad	503	3256	8	0.477
UGrad	1220	43,208	5	0.298
HS	10,296	54,654	12	0.122
SC	5523	82,656	5	0.200
DM	15,326	486,970	7	0.290

Table 1: Statistics for each of the real network datasets that we consider.

5 Experiments and Discussion

The primary purpose of our experiments is to demonstrate that real networks possess multiple layers of meaningful community structure, and that this structure is not detected by any of the other popular community detection algorithms that we consider.

To this end, our experiments fall into four major categories.

1. In Section 5.3, in order to give the reader intuition about the structure of community layers, we present simple synthetic networks based on the stochastic blockmodel template, modified to possess several meaningful community partitionings.

2. In Section 5.4, we perform extensive experiments on real datasets from different domains. We show that these datasets often contain many layers of community structures with high modularity scores, and that these layers, though meaningful, are generally not detected by any of the comparison algorithms that we consider.

3. In Section 5.5, we provide a case study on a small social network containing multiple layers of annotated community structure. HICODE is able to discover the hidden second annotated layer, while comparison algorithms are unable.

4. In Sections 5.6 and 5.7, we quantify the contributions of the identification and refinement steps, and analysis the complexity of HICODE.

These results, taken as a whole, show two main results:

1. HICODE typically finds several high-modularity layers of network communities. Other algorithms are unable to find these deeper layers.
2. When the network dataset contains multiple layers of ground truth communities (where each layer partitions the nodes), HICODE is able to discover the weaker layers much better than other algorithms.

5.1 Preliminaries For each of these sets of experiments, we consider three versions of HICODE. As base community detection algorithms, we use the partitioning algorithms Infomap, the Louvain method, and Walktrap. We refer to these three implementations as HICODE:IM, HICODE:Mod, and HICODE:WT.

We compare HICODE to the overlapping community detection algorithms OSLOM, Link Communities (LC), Greedy Clique Expansion (GCE)[5], and Cascade[6]. This latter algorithm is the closest in spirit to HICODE. We also apply the partitioning algorithms Infomap (IM), Louvain method for greedy modularity optimization (Mod), and Walktrap (WT).[7]

To quantify an algorithm's performance, we compare its output to a set of ground truth communities if we know all the ground truth (the synthetic data), or the ground truth communities have a comparatively high value on modularity (UGrad). We adapt the standard definition of precision, recall, and F1 score in the area of community detection by finding the best match community for a given community basing on value of their Jaccard similarity.

Definition 1. *JaccardCommunityPrecision (JCPrecision): For each detected community C_D, we find the ground truth community C_G that has the greatest Jaccard similarity $J_{D,G}$ to C_D. We calculate these Jaccard scores for every detected community, and define WC-Precision to be the weighted average of these scores, with the weights defined by the size of C_D.*

Definition 2. *JaccardCommunityRecall (JCRecall): We define JCRecall in a similar fashion as JCPrecision, except that we iterate over every ground truth community C_G, and find the most similar detected community C_D, rather than vice versa.*

Definition 3. *JaccardCommunityF1 (JCF1): The harmonic mean of JCPrecision and JCRecall.*

We use this definition rather than NMI because JCPrecision and JCRecall are more specific, telling us how many detected communities are relevant, and how many true communities were found. Weighting these scores by community size gives more importance to larger communities.

5.2 Selecting a HICODE Implementation We performed exeriments using all three reduction methods

[5] GCE use a minimum-clique-size parameter. A value of 4 for the real datasets and 3 for the synthetic datasets worked best.

[6] In [18], Cascade was implemented using Link Communities and Clique Percolation as base algorithms. We present results for Link Communities, which performed better.

[7] Walktrap uses a parameter governing the length of the random walk, and we use the default of 4.

described in Section 3.1 (RemoveEdge, ReduceEdge, and ReduceWeight). In general, ReduceWeight is the best. This occurs because RemoveEdge performs reduction in a very heavy-handed manner by removing all intra-community edges. ReduceWeight, which deterministically reduces the weight of intra-community edges, outperforms ReduceEdge, which randomly removes intra-community edges; however, ReduceWeight can only be used when the base algorithm allows for weighted networks. When this is not the case, ReduceEdge is the usually the best reduction method. Of the three base algorithms we consider, the Louvain method and Walktrap allow for weighted networks. To save space, rather than reporting all results, we present results for HICODE:Mod and HICODE:WT using the ReduceWeight reduction method, and for HICODE:IM using the ReduceEdge reduction method.

5.3 Evaluation on Synthetic Data A layer of communities is hidden because its structure is not strong enough to be observed through that of the dominant layer of communities. This may happen, for example, because the hidden communities are sparser than those in the dominant layers. To demonstrate how layers of communities might be hidden, and illustrate how HICODE identifies these hidden layers, we present two simple synthetic networks that are based on the stochastic blockmodel generative model. Each of these networks has 3000 nodes and at least two layers of planted communities, where each layer corresponds to a single stochastic blockmodel network.

More specifically, each layer is formed by partitioning the nodes into roughly equally-sized sets. We first create the appropriate number of community IDs, and then randomly assign each node to a community. We produce a $G(n, p)$ Erdos-Rnyi random graph over each of these communities (e.g., the 3000 nodes may be partitioned into 100 sets of approximately 30 nodes, and a $G(n, p)$ graph is generated over each of these sets). We select p so that the modularity scores of each layer are similar.

We generate two synthetic networks:

- **SynL2**: SynL2 contains 2 layers of communities, each represents a random partitioning of the 3,000 nodes in the network. The first layer of communities contains 100 communities of size 30 and the second layer contains 50 communities of size 60. Edges within each community are added with probability p, where the p value for the first layer is 0.16 and the p value for the second layer is 0.08. There are a total of $14,446$ edges. The modularity scores for layers 1 and 2 are, respectively, 0.491 and 0.492.
- **SynL3**: This network is generated by adding a third layer to SynL2. This third layer has 30 communities of size around 100, and edges within each community in this layer are added with probability $p = 0.048$. This network has a total of $21,510$ edges. The modularity scores of the three layers in SynL3 are 0.332, 0.321, and 0.323.

Each of these networks contains mutiple community layers of roughly equal strengths (modularity scores). We will demonstrate that although most community detection algorithms can find one layer (usually the layer containing the smallest, densest communities), the other layers are hidden from detection.

Table 2 contains the JCPrecision, JCRecall, and JCF1 scores of HICODE and the comparison algorithms applied to these networks.

On network **SynL2**, when assessing how well the algorithms discover the entire set of planted communities, the HICODE implementations achieve almost perfect JCF1 scores. OSLOM, the best comparison algorithm, has a JCF1 score of only 0.695.

On **SynL3**, we see similar results, though because this network is more complex, JCF1 scores are slightly reduced. Again, the three HICODE implementations are substantially better than the other algorithms.

We now examine these results in more detail. We first note that HICODE returns the correct number of community layers on **SynL3**, and these layers each correspond to a layer of planted communities. Results showing the similarity between the layers found by HICODE and the planted layers are in Table 3. All three HICODE implementations return layers that are clearly associated with the planted layers. The third, sparser layer of planted communities is hardest for other algorithms to discover. Results are similar for **SynL2**.

These experiments on simple synthetic data demonstrate two important points. First, existing algorithms are unable to locate communities outside the dominant layer, even if those communities are meaningful (e.g., a high modularity score). Second, by removing the effects of dominant community layers, HICODE is able to achieve much higher scores than competing methods.

5.4 Evaluation on Real Data We now show that real networks from a variety of domains contain hidden community structure. These layers individually have high modularity scores, indicating that they are significant. However, because some of these layers are hidden behind stronger, more dominant layers of communities, our comparison community detection algorithms are unable to discover them.

Our argument contains three parts: First, we use HICODE to show that the networks contain multiple high-modularity partitions. Second, we compare these

		HICODE			Overlapping				Partitioning		
		HC:Mod	**HC:IM**	**HC:WT**	Cascade	OSLOM	LC	GCE	Mod	IM	WT
SynL2	JCRecall	0.974	0.986	0.949	0.348	0.651	0.288	0.495	0.183	0.475	0.190
	JCPrecision	0.975	0.985	0.950	0.184	0.744	0.220	0.473	0.389	0.847	0.273
	JCF1	0.975	0.986	0.950	0.240	0.695	0.249	0.484	0.249	0.609	0.224
SynL3	JCRecall	0.945	0.971	0.908	0.260	0.464	0.212	0.329	0.107	0.312	0.120
	JCPrecision	0.948	0.965	0.914	0.249	0.671	0.171	0.340	0.313	0.813	0.223
	JCF1	0.947	0.968	0.911	0.254	0.548	0.189	0.335	0.159	0.451	0.156

Table 2: JCRecall, JCPrecision, and JCF1 scores of all algorithms when evaluated on synthetic data. HC:Mod, HC:WT, and HC:IM refer to the three implementations of HICODE corresponding to different base algorithms. Note that all HICODE implementations substantially outperform the other algorithms for finding overlapping communities. HICODE:Mod, HICODE:WT, and HICODE:IM refer to the three implementations of HICODE corresponding to different base algorithms.

		HICODE									Overlaping				Partitioning		
		HC:Mod			**HC:IM**			**HC:WT**									
		Layer 1	Layer 2	Layer 3	Layer 1	Layer 2	Layer 3	Layer 1	Layer 2	Layer 3	Cascade	OSLOM	LC	GCE	Mod	IM	WT
PL1	JCRec.	0.905	0.034	0.029	0.970	0.034	0.031	0.826	0.034	0.030	0.511	0.667	0.419	0.564	0.219	0.838	0.200
	JCPrec.	0.914	0.037	0.035	0.965	0.036	0.035	0.843	0.037	0.035	0.239	0.410	0.160	0.259	0.313	0.812	0.213
	JCF1	0.909	0.035	0.031	0.968	0.035	0.033	0.834	0.035	0.032	0.326	0.508	0.231	0.355	0.258	0.825	0.258
PL2	JCRec.	0.037	0.966	0.035	0.037	0.974	0.035	0.038	0.945	0.035	0.181	0.481	0.146	0.293	0.050	0.056	0.093
	JCPrec.	0.035	0.966	0.038	0.033	0.972	0.037	0.035	0.947	0.037	0.075	0.246	0.053	0.124	0.060	0.040	0.107
	JCF1	0.036	0.966	0.036	0.035	0.973	0.036	0.036	0.946	0.036	0.106	0.326	0.078	0.174	0.054	0.047	0.099
PL3	JCRec.	0.036	0.038	0.965	0.035	0.037	0.967	0.037	0.038	0.953	0.088	0.243	0.070	0.131	0.051	0.041	0.068
	JCPrec.	0.031	0.035	0.965	0.029	0.035	0.959	0.032	0.035	0.952	0.042	0.102	0.028	0.055	0.055	0.031	0.070
	JCF1	0.033	0.037	0.965	0.032	0.036	0.963	0.034	0.037	0.952	0.057	0.144	0.040	0.078	0.053	0.035	0.069

Table 3: JCRecall, JCPrecision, and JCF1 scores of all algorithms when evaluated on the synthetic SynL3 community layers. PL1, PL2, and PL3 are the three layers of planted communities, with PL1 containing small, dense communities and PL3 containing larger, sparse communities. The layers found by HICODE are strongly associated with the three planted layers. Other algorithms cannot find the sparse PL3 layer.

layers to one another, demonstrating that they are non-redundant. Finally, we use the JCRecall metric, defined in Definition 2, to show that these hidden communities are not detected by the comparison algorithms.

First, to show that these networks contain multiple hidden layers, we apply HICODE to the real networks described in Section 4. Table 4 shows the number of layers found by HICODE:Mod on each network, and their modularity scores. We see similar results with the other HICODE implementations. HICODE found multiple layers on every network, and some lower layers have modularity scores nearly as high as the first layer. For example, HICODE found 4 layers on network HS, and while the first layer has the highest modularity value, many of the others are close behind.

We have shown that these real networks contain multiple layers of meaningful community structure, and we now demonstrate that these layers are non-redundant. Table 4 contains the maximum Normalized Mutual Information between each pair of layers found by HICODE:Mod on the various networks. For most networks, the NMI values are very low, indicating that the layers are distinct, even in cases such as SC where a large number of layers were found. These results are typical of other HICODE implementations.

We have just seen that the real networks have multiple dissimilar, meaningful layers, and we now demonstrate that these layers are not found by other algorithms. To measure this, we use the JCRecall measure from Definition 2 to evaluate the similarity

Network	**# Layers**	**Max. NMI**	**Modularity Scores of Layers**
synL2	2	0.20	0.49, 0.49
synL3	3	0.20	0.34, 0.32, 0.32
Grad	2	0.40	0.64, 0.57
UGrad	3	0.07	0.39, 0.27, 0.20
HS	4	0.15	0.41 0.31 0.36 0.27
SC	6	0.10	0.33 0.22 0.17 0.22 0.18 0.11
DM	3	0.19	0.35 0.25 0.24

Table 4: Number of Layers found by HICODE:Mod on each network, as well as modularity scores of each layer. These modularity scores are calculated in the original, not the reduced, graph. Note that there are often high-modularity partitions beyond the first layer.

	Overlapping				Partitioning		
	Cascade	OSLOM	LC	GCE	Mod	IM	WT
UGrad L1	0.43	0.38	0.43	0.83	0.85	0.89	0.58
UGrad L2	0.13	0.13	0.11	0.18	0.14	0.14	0.34
UGrad L3	0.15	0.12	0.14	0.20	0.16	0.15	0.17

Table 5: JCRecall scores measuring how well each algorithm discovered the communities found in the different layers of network UGrad found by HICODE:Mod. The other algorithms find the first layer of UGrad very well, but the JCRecall scores quickly go down.

between the deeper layers found by HICODE and the outputs of other algorithms. If the JCRecall is low, then we conclude that the other algorithms did not find the communities in that layer.

Table 5 shows how well other algorithms locate the communities found by HICODE:Mod on network **UGrad** using JCRecall. Other algorithms do very well at identifying the first layer of HICODE communities, but do poorly at identifying the hidden layers. Results for other networks are similar.

We thus conclude that the layers found by HICODE have high modularity and are meaningful, but are not detected by other algorithms.

5.5 Case Study We now study the real network UGrad in detail. This network is especially interesting because it contains ground truth community annotation; moreover, unlike the multitudes of other networks that contain community annotation, the annotated structure here is *layered*. This network of students contains three types of annotation: for each student, we know the **Department**, **Dormitory**, and **Year** of that student. Each of these categories corresponds to a partitioning of the network (e.g., every student has one department).

The modularity score for the Department category is only 0.05, and we expect that none of the algorithms will discover this layer. The modularity scores for Year and Dormitory are, respectively, 0.26 and 0.38. We expect that most algorithms will be able to find the stronger Dormitory communities, but that the Year communities will only be apparent once the stronger Dormitory structure has been reduced.

We apply the three HICODE implementations to UGrad. The last row of Table 6 shows the JCF1 scores of HICODE and other algorithms evaluated on the task of discovering the ground truth communities.

First, note that all HICODE implementations outperform OSLOM, LC, and GCE by a very large amount. Two HICODE implementations outperform all partitioning methods, while the third (HICODE:IM) is slightly outperformed by greedy modularity optimization. It is interesting that the partitioning methods perform so well. This is because these methods find a few communities almost perfectly, while the overlapping methods find more communities, but less accurately.

HICODE's most remarkable behavior is seen by comparing its output to the ground truth categories.

The different HICODE implementations identify either two or three layers of communities, and in every case, two of these detected layers very strongly correspond to two ground truth categories. No algorithm is able to identify the department communities, which have a very low modularity score. Table 6 contains results for each of the algorithms evaluated against the Dormitory and Year ground truth communities.

For each HICODE implementation, one layer is clearly associated with the Dormitory communities, and one with the Year communities. Notably, *none* of the other algorithms does a good job at discovering these 'Year' communities, because these communities are only visible once the stronger layer has been eliminated.

5.6 Identification vs. Refinement Recall that in the refinement stage, for each detected layer, the effects of *all other* layers are reduced, and the layer under consideration is re-detected. In contrast, the identification stage reduces the effects of only the *stronger* layers. However, accurate detection of each layer is hindered by both the stronger layers and the weaker layers.

For the four datasets for which we have a ground truth (SynL2, SynL3, Grad, and UGrad), we use the JCF1 measure described earlier to calculate how well the communities found by HICODE correspond to the ground truth. We perform this calculation immediately after the identification stage before refinement, as well as after 30 iterations of refinement. Table 7 shows how much the refinement process improved upon the initial JCF1 scores found after the identification. In general, refinement dramatically improves the initial results.

5.7 Running Time HICODE's running time depends on the running time of the base algorithm. Formally, the running time of HICODE is $O(L^2 * k * A)$, where L is the number of layers detected, k is the number of refinement iterations performed, and A is the running time of the base algorithm. Because HICODE selects the number of layers by trying all possible numbers up to the final selection L, this expression contains an L^2 term. The cost of reducing community structure is small compared to the cost of the base algorithm.

6 Conclusions

Traditional paradigms view communities as disjoint, overlapping, or hierarchical sets. We argue that the complementary concept of hidden layered structure is necessary for understanding networks. We demonstrated that real networks may contain several layers of communities, with weaker layers obscured by more dominant structure. Our algorithm, HICODE, can identify these layers of structure by iteratively removing stronger layers to reveal the weaker structure beneath.

We hope that our work serves as a starting point for future research in community structure and community detection. Many open questions remain. In the short term, we are interested in extended HICODE to accom-

		HICODE						Overlapping				Partitioning		
		HC:Mod		**HC:IM**		**HC:WT**								
		Layer 1	Layer 2	Layer 1	Layer 2	Layer 1	Layer 2	Cascade	OSLOM	LC	GCE	Mod	IM	WT
Dorm	JCRec.	0.896	0.102	0.849	0.111	0.802	0.113	0.430	0.378	0.428	0.797	0.800	0.853	0.527
	JCPrec.	0.896	0.100	0.786	0.117	0.764	0.128	0.102	0.262	0.093	0.665	0.789	0.789	0.492
	JCF1	0.896	0.101	0.817	0.114	0.783	0.120	0.165	0.309	0.153	0.725	0.795	0.820	0.529
Year	JCRec.	0.100	0.720	0.101	0.286	0.105	0.593	0.130	0.120	0.128	0.123	0.137	0.104	0.270
	JCPrec.	0.102	0.705	0.095	0.321	0.098	0.574	0.031	0.089	0.029	0.106	0.119	0.097	0.170
	JCF1	0.101	0.713	0.098	0.302	0.101	0.584	0.050	0.102	0.048	0.114	0.127	0.100	0.208
Overall	JCF1	0.584		0.470		0.527		0.157	0.236	0.153	0.456	0.467	0.476	0.389

Table 6: JCRecall, JCPrecision, and JCF1 scores of all algorithms on real network UGrad community categories. HC:Mod, HC:WT, and HC:IM refer to the three implementations of HICODE corresponding to different base algorithms. Note that the layers found by HICODE implementations are very strongly associated with two UGrad community categories. Other algorithms have difficulty finding the hidden 'Year' layer. The final line shows the overall performance of the different algorithms at discovering all ground truth communities (which includes communities other than the 'Dorm' and 'Year' communities). HICODE outperforms all overlapping methods.

Network	**HICODE:Mod**	**HICODE:IM**	**HICODE:WT**
SynL2	25% (0.99)	4% (0.99)	68% (0.99)
SynL3	127% (0.97)	17% (0.98)	210% (0.97)
Grad	5% (0.59)	0% (0.55)	-3% (0.60)
UGrad	21% (0.78)	0% (0.42)	21% (0.67)

Table 7: Average improvement of NMI score from the initial layers detected after the first identification step, to the final output after refinement steps are conducted. Values in parentheses indicate average NMI similarity between detected layers and ground truth after refinement.

modate overlapping community detection algorithms as the base method. Other interesting problems include developing other approaches for identifying such hidden structure, which do not remove dominant community structure. We are also pursuing a deeper understanding of the causes behind hidden structure: why are some communities hidden? Finally, we hope to develop a theoretical foundation to characterize hidden structure.

References

[1] Y.-Y. Ahn, J. P. Bagrow, and S. Lehmann. Link communities reveal multiscale complexity in networks. *Nature*, 466:761–764, 2010.

[2] V. D. Blondel, J.-L. Guillaume, R. Lambiotte, and E. Lefebvre. Fast unfolding of communities in large networks. *Journal of Statistical Mechanics: Theory and Experiment*, 10, 2008.

[3] P.-Y. Chen and A. Hero. Deep community detection. *CoRR*, abs/1407.6071, 2014.

[4] S. Fortunato. Community detection in graphs. *Physics Reports*, 486:75–174, 2010.

[5] S. Fortunato and C. Castellano. Community structure in graphs. *Computational Complexity*, pages 490–512, 2012.

[6] M. Girvan and M. E. Newman. Community structure in social and biological networks. *Proceedings of the National Academy of Sciences*, 99(12):7821–7826, 2002.

[7] A. Lancichinetti, F. Radicchi, J. J. Ramasco, and S. Fortunato. Finding statistically significant communities in networks. *PLoS One*, 6(4), 2011.

[8] C. Lee, F. Reid, A. McDaid, and N. Hurley. Detecting highly overlapping community structure by greedy clique expansion. In *SNAKDD Workshop*, 2010.

[9] X. Liu, Y. Zhou, C. Hu, and X. Guan. Detecting community structure for undirected big graphs based on random walks. In *WWW*, 2014.

[10] A. Mislove, B. Viswanath, K. P. Gummadi, and P. Druschel. You are who you know: Inferring user profiles in online social networks. In *WSDM*, pages 251–260, 2010.

[11] M. E. Newman. Modularity and community structure in networks. *Proceedings of the National Academy of Sciences*, 103(23):8577–8582, 2006.

[12] L. Pan, C. Dai, C. Wang, and J. Xie. Overlapping community detection via leader-based local expansion in social networks. In *ICTAI*, 2012.

[13] P. Pons and M. Latapy. Computing communities in large networks using random walks. *Journal of Graph Algorithms and Applications*, 10(2):191–218, 2006.

[14] M. Rosvall and C. T. Bergstrom. Maps of random walks on complex networks reveal community structure. *Proceedings of the National Academy of Sciences*, 105(4):1118–1123, 2008.

[15] W. Wang, D. Liu, X. Liu, and L. Pan. Fuzzy overlapping community detection based on local random walk and multidimensional scaling. *Physica A*, pages 6578–6586, 2013.

[16] J. J. Whang, D. F. Gleich, and I. S. Dhillon. Overlapping community detection using seed set expansion. In *CIKM*, 2013.

[17] J. Xie, S. Kelley, and B. K. Szymanski. Overlapping community detection in networks: The state-of-the-art and comparative study. *ACM Computing Surveys*, 45(4), 2013.

[18] J.-G. Young, A. Allard, L. Hebert-Dufresne, and L. J. Dube. Unveiling hidden communities through cascading detection on network structures. *CoRR*, abs/1211.1364, 2012.